UNIT-II

Software Requirements: Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

Requirements engineering process: Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management.

System models: Context models, behavioral models, data models, object models, structured methods.

SOFTWARE REQUIREMENTS

Software requirements are necessary

1. To introduce the concepts of user and system requirements
2. To describe functional and non-functional requirements
3. To explain how software requirements may be organized in a requirements document

What is a requirement?

1. The requirements for the system are the description of the services provided by the system and its operational constraints
    1. It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
    2. This is inevitable as requirements may serve a dual function
    3. May be the basis for a bid for a contract - therefore must be open to interpretation;
    4. May be the basis for the contract itself - therefore must be defined in detail; Both these statements may be called requirements

Requirements engineering:

1. The process of finding out, analysing documenting and checking these services and constraints is called requirement engineering.
2. The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
    1. The requirements themselves are the descriptions ofthe system services and constraints that are generated during the requirements engineering process.

Requirements abstraction (Davis):

If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The **requirements** must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs. Once a contract has been awarded, the contractor must

write a **system definition** for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the **requirements document** for the system."

Types of requirement:

1. **User requirements**

Statements in natural language plus diagrams ofthe services the system provides and its operational constraints. Written for customers.

2. **System requirements**

A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

User Requirement Definition:

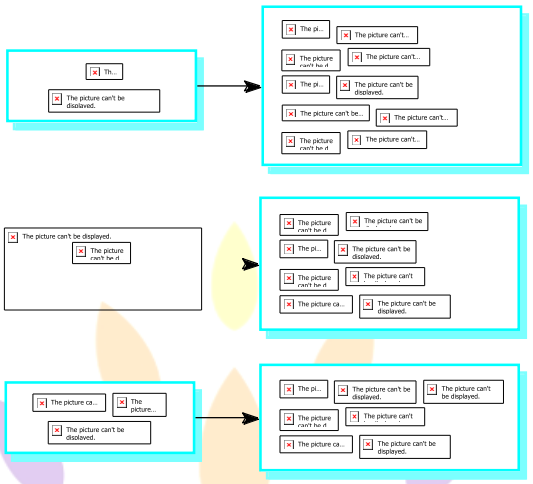
The software must provide the means of representing and accessing external files created by other

tools.

System Requirement specification:

1. The user should be provided with facilities to define the type of external files.
2. Each external file type may have an associated tool which may be applied to the file.
3. Each external file type may be represented as a specific icon on the user's display.
4. Facilities should be provided for the icon representing an external file type to be defined bythe user.
5. When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Requirements readers:



1.Functional and non-functional requirements: Functional requirements

Statements of services the system should provide how the system should react to particular inputs and how the system should behave in particularsituations.

Non-functional requirements

Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

Domain requirements

Requirements that come from the application domain of the system and that reflect characteristics of that domain.

1.1 FUNCTIONAL REQUIREMENTS:

Describe functionality or system services.

Depend on the type of software, expected users and the type of system where the softwareis used.

Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

The functional requirements for **The LIBSYS system:**

A library system that provides a single interface to a number of databases of articles in different libraries. Users can search for, download and print these articles for personal study.

Examples of functional requirements

1. The user shall be able to search either all ofthe initial set of databases or select a subset from it.
2. The system shall provide appropriate viewers for the user to read documents in thedocument store.
3. Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storagearea.

Requirements imprecision

1. Problems arise when requirements are not precisely stated.
2. Ambiguous requirements may be interpreted in different ways by developers and users.
3. Consider the term 'appropriate viewers'
4. User intention - special purpose viewer for each different document type;
5. Developer interpretation - Provide a text viewer that shows the contents of the document.

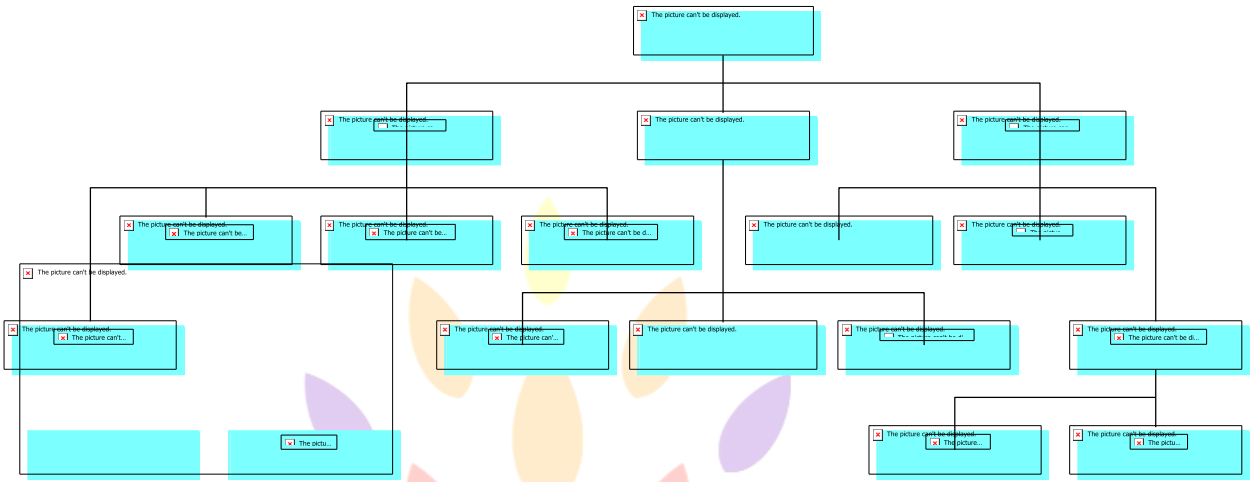
Requirements completeness and consistency:

In principle, requirements should be both complete and consistent. Complete

They should include descriptions of all facilities required. Consistent
There should be no conflicts or contradictions in the descriptions of the system facilities. In practice, it is impossible to produce a complete and consistent requirements document.

NON-FUNCTIONAL REQUIREMENTS

1. These define system properties and constraints e.g. reliability, response time andstorage requirements. Constraints are I/O device capability, system representations, etc.
2. Process requirements may also be specified mandating a particular CASE system, programming language or development method.
3. Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.



Non-functional requirement types:

Non-functional requirements: Product requirements

1. Requirements which specify that the delivered product must behave in a particular way

   e.g. execution speed, reliability, etc.

USER REQUIREMENTS

2. Should describe functional and non-functional requirements in such a way that they are understandable by system users who don't have detailed technicalknowledge.
3. User requirements are defined using natural language, tables and diagrams as these can be understood by all users.

Problems with natural language

Lack of clarity

1. Precision is difficult without making the document difficult to read.

Requirements confusion

2. Functional and non-functional requirements tend to bemixed-up.

Requirements amalgamation

3. Several different requirements may be expressed together.

Requirement problems

Database requirements includes both conceptual and detailed information

Describes the concept of a financial accounting system that is to be included in LIBSYS; However, it also includes the detail that managers can configure this system - this is unnecessary at this level.

Grid requirement mixes three different kinds of requirement

1. Conceptual functional requirement (the need for a grid);
2. Non-functional requirement (grid units);
3. Non-functional UI requirement (grid switching).
4. Structured presentation

Guidelines for writing requirements

Invent a standard format and use it for all requirements.

1. Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
2. Use text highlighting to identify key parts of the requirement.
3. Avoid the use of computer jargon.

SYSTEM REQUIREMENTS

1. More detailed specifications of system functions, services and constraints than userrequirements.
2. They are intended to be a basis for designing the system.
3. They may be incorporated into the system contract.
4. System requirements may be defined or illustrated using system models

Requirements and design

In principle, requirements should state what the system should do and the design should describe how it does this.

In practice, requirements and design are inseparable

1. A system architecture may be designed to structure the requirements;
2. The system may inter-operate with other systems that generate design requirements;
3. The use of a specific design may be a domain requirement.

Problems with NL (natural language) specification

Ambiguity

The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult.

Over-flexibility

The same thing may be said in a number of different ways in the specification. Lack of modularization.
NL structures are inadequate to structure system requirements.

Alternatives to NL specification:

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of thesystem. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used.
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

Structured language specifications

1. The freedom of the requirements writer is limited by a predefined template for requirements.
2. All requirements are written in a standard way.
3. The terminology used in the description may belimited.

4. The advantage is that the most of the expressiveness of natural language is maintained but a degree of uniformity is imposed on the specification.

Form-based specifications

5. Definition of the function or entity.
6. Description of inputs and where they come from.
7. Description of outputs and where they go to.
8. Indication of other entities required.
9. Pre and post conditions (if appropriate).
10. The side effects (if any) of the function.

Tabular specification

11. Used to supplement natural language.
12. Particularly useful when you have to define a number of possible alternative courses of action.

Graphical models

13. Graphical models are most useful when you need to show how state changes or where you need to describe a sequence of actions.

Sequence diagrams

1. These show the sequence of events that take place during some user interaction with a system.
2. You read them from top to bottom to see the order of the actions that take place.
3. Cash withdrawal from an ATM

1. Validate card;
2. Handle request;
3. Complete transaction.

Sequence diagram of ATM withdrawal

System requirement specification using a standard form:

1. Function
2. Description
3. Inputs
4. Source
5. Outputs
6. Destination

- -

7. Action
8. Requires
9. Pre-condition
10. Post-condition
11. Side-effects

When a standard form is used for specifying functional requirements, the following information should be included:

1. Description of the function or entity beingspecified
2. Description of its inputs and where these comefrom
3. Description of its outputs and where these go to
4. Indication of what other entities are used
5. Description of the action to be taken
6. If a functional approach is used, a pre-condition setting out what must be true before the function is called and a post-condition specifying what is true after the function is called
7. Description of the side effects of the operation.

1. INTERFACE SPECIFICATION

1. Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.
2. Three types of interface may have to be defined
   1. ***Procedural interfaces*** where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. These interfaces are sometimes called Applicatin Programming Interfaces (APIs)
   2. ***Data structures that are exchanged*** that are passed from one sub-system to another. Graphical data models are the best notations for this type of description
   3. ***Data representations*** that have been established for an existing sub-system
3. Formal notations are an effective technique for interface specification.

4. THE SOFTWARE REQUIREMENTS DOCUMENT:

1. The requirements document is the official statement of what is required of the system developers.

2. Should include both a definition of user requirements and a specification of the system requirements.
3. It is NOT a design document. As far as possible, it should set of WHAT the system shoulddo rather than HOW it should do it

Users of a requirements document:

IEEE requirements standard defines a generic structure for a requirements document that must be instantiated for each specific system.

1. Introduction.
   1. Purpose of the requirements document
   2. Scope of the project
   3. Definitions, acronyms and abbreviations
   4. References
   5. Overview of the remainder of the document
2. General description.
   1. Product perspective
   2. Product functions
   3. User characteristics
   4. General constraints
   5. Assumptions and dependencies

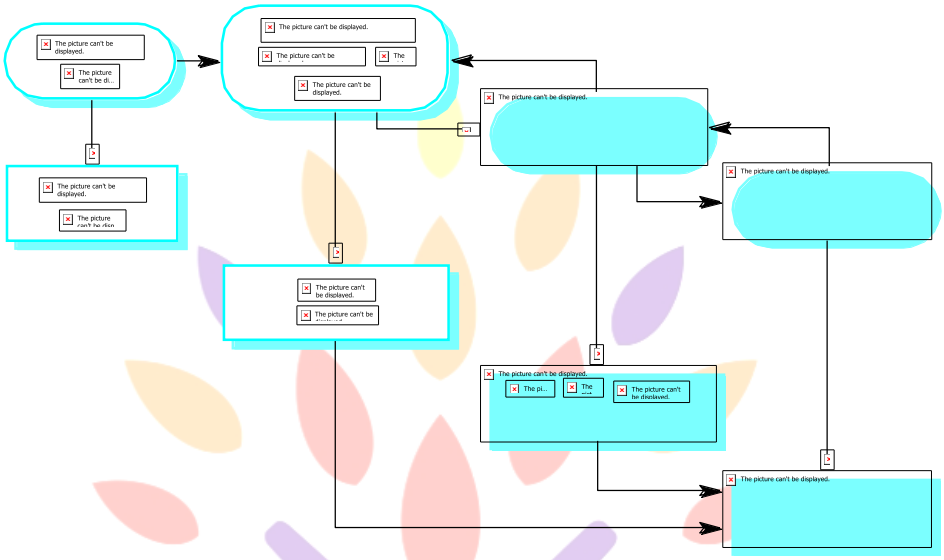
3. Specific requirements cover functional, non-functional and interface requirements. The requirements may document external interfaces, describe system functionality and performance, specify logical database requirements, design constraints, emergent system properties and quality characteristics.
4. Appendices.
5. Index.

REQUIREMENTS ENGINEERING PROCESSES

The **goal** of requirements engineering process is to create and maintain a system requirements document. The overall process includes four high-level requirement engineering sub-processes. These are concerned with

1. Assessing whether the system is useful to the business(feasibility study)
2. Discovering requirements(elicitation and analysis)
3. Converting these requirements into some standard form(specification)
4. Checking that the requirements actually define the system that the customerwants(validation) The process of managing the changes in the requirements is called **requirementmanagement**.

The requirements engineering process



Requirements engineering:

The alternative perspective on the requirements engineering process presents the process as a **three-stage activity** where the activities are organized as an iterative process around a spiral. The amount of time and effort devoted to each activity in iteration depends on the stage of the overall process and the type of system being developed. Early in the process, most effort will be spent on understanding high-level business and non- functional requirements and the user requirements. Later in the process, in the outer rings of the spiral, more effort will be devoted to system requirements engineering and systemmodeling.

This spiral model accommodates approaches to development in which the requirements are developed to different levels of detail. The number of iterations around the spiral can vary, so the spiral can be exited after some or all of the user requirements have been elicited.

Some people consider requirements engineering to be the process of applying a structured analysis method such as object-oriented analysis. This involves analyzing the system and

developing a set of graphical system models, such as use-case models, that then serve as a system specification. The set of models describes the behavior of the system and are annotated with additional information describing, for example, its required performance or reliability.

1. FEASIBILITY STUDIES

A feasibility study decides whether or not the proposed system is worthwhile. The input to the feasibility study is a set of preliminary business requirements, an outline description of the system and how the system is intended to support business processes. The results of the feasibility study should be a report that recommends whether or not it worth carrying on with the requirements engineering and system development process.

1.A short focused study that checks
1.If the system contributes to organizational objectives;
2.If the system can be engineered using current technology and withinbudget;
3.If the system can be integrated with other systems that are used.

Feasibility study implementation:

2. A feasibility study involves information assessment, information collection and report writing.
3. Questions for people in the organization
1.What if the system wasn't implemented?
2.What are current process problems?
3.How will the proposed system help?
4.What will be the integration problems?
5.Is new technology needed? What skills?
6. What facilities must be supported by the proposed system?

In a feasibility study, you may consult information sources such as the managers of the departments where the system will be used, software engineers who are familiar with the type of system that is proposed, technology experts and end-users of the system. They should try to complete a feasibility study in two or three weeks.

Once you have the information, you write the feasibility study report. You should make a recommendation about whether or not the system development should continue. In the report, you may propose changes to the scope, budget and schedule of the system and suggest further high-level requirements for the system.

4.REQUIREMENT ELICITATION AND ANALYSIS:

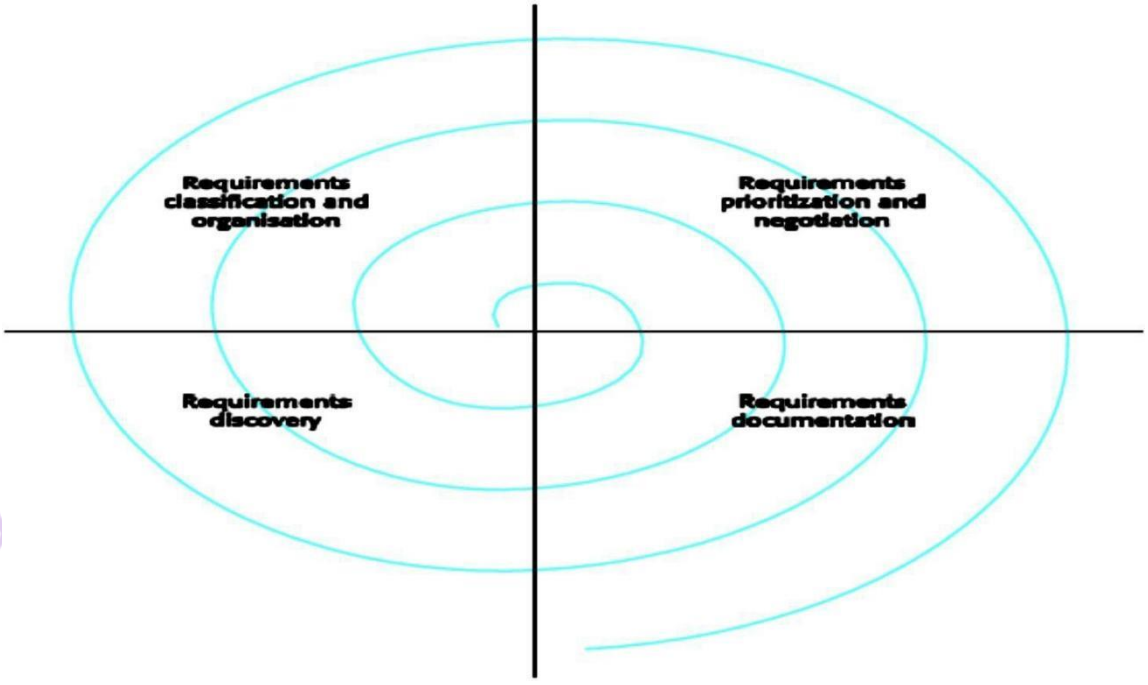
The requirement engineering process is requirements elicitation and analysis.

5. Sometimes called requirements elicitation or requirements discovery.
6. Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
7. May involve end-users, managers, engineers involved in maintenance, domainexperts, trade unions, etc. These are called *stakeholders.*

Problems of requirements analysis

8. Stakeholders don't know what they really want.
9. Stakeholders express requirements in their own terms.
10. Different stakeholders may have conflicting requirements.
11. Organizational and political factors may influence the system requirements.
12. The requirements change during the analysis process. New stakeholders mayemerge and the business environment change.

The requirements spiral



Process activities

1. Requirements discovery

Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.

2. Requirements classification and organization

Groups related requirements and organizes them into coherent clusters.

3. Prioritization and negotiation

Prioritizing requirements and resolving requirements conflicts.

4. Requirements documentation

Requirements are documented and input into the next round of the spiral.

The process cycle starts with requirements discovery and ends with requirements documentation. The analyst's understanding of the requirements improves with each round of the cycle.

Requirements classification and organization is primarily concerned with identifying overlapping requirements from different stakeholders and grouping related requirements. The most common way of grouping requirements is to use a model of the system architecture to identify subsystems and to associate requirements with each sub-system.

Inevitably, stakeholders have different views on the importance and priority of requirements, and sometimes these view conflict. During the process, you should organize regular stakeholder negotiations so that compromises can be reached.

In the requirement documenting stage, the requirements that have been elicited are documented in such a way that they can be used to help with further requirements discovery.

5.REQUIREMENTS VALIDATION

1. Concerned with demonstrating that the requirements define the system that the customer really wants.
2. Requirements error costs are high so validation is very important
3. Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking:

1. ***Validity***: Does the system provide the functions which best support the customer's needs?
2. ***Consistency***: Are there any requirements conflicts?
3. ***Completeness***: Are all functions required by the customer included?
4. ***Realism***: Can the requirements be implemented given available budget and technology
5. ***Verifiability***: Can the requirements be checked?

Requirements validation techniques

1. Requirements reviews
2. Systematic manual analysis of the requirements.
3. Prototyping
4. Using an executable model of the system to check requirements. Covered in Chapter 17.
5. Test-case generation
6. Developing tests for requirements to check testability.

Requirements reviews:

1. Regular reviews should be held while the requirements definition is being formulated.
2. Both client and contractor staff should be involved in reviews.
3. Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an earlystage.

Review checks:

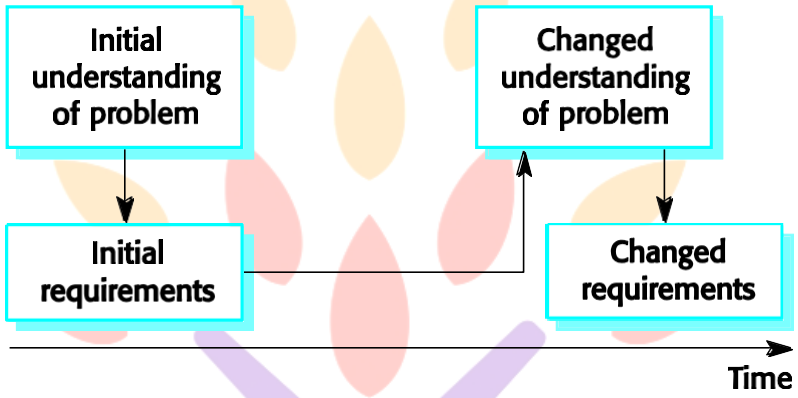
1. ***Verifiability*:** Is the requirement realistically testable?
2. ***Comprehensibility*:** Is the requirement properly understood?
3. ***Traceability*:** Is the origin of the requirement clearly stated?
4. ***Adaptability*:** Can the requirement be changed without a large impact on other requirements?

5.REQUIREMENTS MANAGEMENT

1. Requirements management is the process of managing changing requirements duringthe requirements engineering process and system development.
2. Requirements are inevitably incomplete and inconsistent
3. New requirements emerge during the process as business needs change and abetter understanding of the system is developed;
4. Different viewpoints have different requirements and these are oftencontradictory.
5. **Requirements change**
6. The priority of requirements from different viewpoints changes during the developmentprocess.

7. System customers may specify requirements from a business perspective that conflict withend-user requirements.
8. The business and technical environment of the system changes during its development.

Requirements evolution:



Enduring and volatile requirements:

1. ***Enduring requirements***: Stable requirements derived from the core activity of the customer organisation. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
2. ***Volatile requirements***: Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

Requirements classification:

Requirement Type	Description
Mutable requirements	Requirements that change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.

Emergent requirements	Requirements that emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.
Consequential requirements	Requirements that result from the introduction of the computer system. Introducing the computer system may change the organisations processes and open up new ways of working which generate new system requirements
Compatibility requirements	Requirements that depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.

Requirements management planning:

1. During the requirements engineering process, you have to plan:
2. Requirements identification
3. How requirements are individually identified;
4. A change management process
5. The process followed when analysing a requirements change;
6. Traceability policies
7. The amount of information about requirements relationships that is maintained;
1. CASE tool support
2. The tool support required to help manage requirements change;

Traceability:

Traceability is concerned with the relationships between requirements, their sources and the system design

1. Source traceability

2. Links from requirements to stakeholders who proposed these requirements;
3. Requirements traceability
4. Links between dependent requirements;
5. Design traceability - Links from the requirements to the design;

CASE tool support:

1. Requirements storage
2. Requirements should be managed in a secure, managed data store.
3. Change management
4. The process of change management is a workflow process whose stages can bedefined and information flow between these stages partially automated.
5. Traceability management
6. –Automated retrieval of the links between requirements.

Requirements change management:

1. Should apply to all proposed changes to the requirements.
2. Principal stages
3. Problem analysis. Discuss requirements problem and propose change;
4. Change analysis and costing. Assess effects of change on other requirements;
5. Change implementation. Modify requirements document and other documents to reflect change

Change management: